Elitism Enhancements for Genetic Algorithm based Network Intrusion Detection System

[1] Tanapuch Wanwarang, [2] Machigar Ongtang
[1]*Dimension Data (Thailand) Limited, E-mail: tanapuch.wanwarang@dimensiondata.com*
[2]*Dhurakij Pundit University, Thailand, E-mail: machigar.ong@dpu.ac.th*

Abstract

Traditional signature-based Network Intrusion Detection Systems (NIDS) suffer from high false negative rate because it can detect an intrusion only if there is an existing rule matching the particular real-time attack. The combination of Genetic Algorithms, network security schemes, and IDS practices has created a model of intelligence system that has the ability to derive new best-fit classification rules from already known attack patterns. Nevertheless, the existing NIDS approaches in this paradigm still experience substantial overhead and limited variety of resulting NIDS rules. We propose in this paper a new genetic algorithm-based NIDS called AceGA, which introduces three novel enhancements namely Wildcard Weight Penalty, Ace Comparison Elimination, and Elite's Traits Induction. It is shown that AceGA provides decreased time overhead and better rules quality. For demonstration purposes, DARPA datasets from MIT Lincoln Lab are used for training and testing the intrusion detection rules. Several simulation experiments are conducted to evaluate the efficiency and effectiveness of each of our proposed enhancements including the overall capability of AceGA to detect our selected types of attacks with satisfactory true positive and false positive rates. Additional statistical results depicting the accuracy, precision, sensitivity, and specificity of the resulting rules are thoroughly analyzed.

Keywords: *Genetic Algorithms, Intrusion Detection System, Network Security, Support-Confidence Framework, Evolutionary Process.*

1. Introduction

Network Intrusion Detection System (NIDS) has been well accepted as a crucial segment for detecting security violations. It monitors network traffic for suspicious packets, malicious activities, or signs of unauthorized access in the traffic of that network, and subsequently alerts the network administrator about such potential intrusions [2, 5]. Signature-based NIDS are widely used by both corporations and home users. Although several features are continually incorporated, the general approach towards signature-based NIDS has not deviated much since its inception; that is, security experts have to carefully and manually construct new classification rules before adding them to the systems [6]. With the increasing number of system vulnerabilities and the endless varieties of network attacks, this NIDS approach suffers high false negativity because it is nearly impossible for the security experts to construct new rules to cover all types of intrusions in time to prevent the attacks [7].

Genetic Algorithms (GA), a well-known subfield of Artificial Intelligence (AI), has promising characteristics for developing NIDS rules. It imitates biological natural evolution using heuristic search to find the optimal solutions to a search problem [8-10]. Studies have shown that GA is adaptive and cost effective for the use as a foundation for NIDS [1, 4], known as GANIDS. However, the prior works on GANIDS still experience several limitations, among which we focus here on three main dimensions. First, the NIDS rules generation process is comparatively time-consuming. Second, individuals in the genetic algorithm evolution naturally evolve themselves following the pattern of existing local maxima. This behavior leads to the lack of rules diversity, which contrasts with NIDS's requirement to detect a broad assortment of attacks. Third, the rules filled with excessive wildcards are mistreated as top-ranked rules. In actuality, they would treat normal network connections as attacks; and therefore, shall be avoided.

In this paper, we explore the three aforementioned challenges in genetic algorithm-based NIDS paradigm and propose an innovative scheme called AceGA. AceGA introduces three novel enhancements to subsequently generate new effective classification rules in a more efficient fashion. First, Ace Comparison Elimination enhancement remedies the local maxima over-fitting problem [4]

and increases the variety of the generated rules. The proposed technique eliminates elite individuals with similar features to only one best elite to preserve the local maxima while ensuring that the second best local maxima is also maintained. Next, Elite's Traits Induction enhancement improves calculation time performance at least by the factor of 4. Lastly, Wildcard Weight Penalty gracefully decreases the score of wildcards filled individual to ensure that the actual global maxima are found in the evolution.

AceGA includes two main modules. The training module allows AceGA to learn from the training data through our enhanced genetic algorithm and accordingly create the best-fit classification NIDS rules for network intrusion detection. The test module leverages the generated NIDS rules to match again test audit data. It represents how AceGA would use the produced rules to detect potential attacks from network connection data in real-world situation. Apparently, the development of AceGA requires appropriate datasets. The quality of the resulting classification rules largely depends on the amount and characteristics of the training data. Suitably for the task, MIT Lincoln Laboratory, under the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, provides the first standard corpora for evaluation of computer network intrusion detection systems, and along with it seven weeks long training data and two weeks test data [11, 12]. These datasets have been used extensively to perform simulation of network intrusion detections to aid security projects and studies [1, 3, 4, 7, 8, 13, 14]. The DARPA evaluation dataset has been found to have the required potential in modeling the attacks that are common in the network connection sessions and is fit for use as the base line of any intrusion detection research [12, 15]. In this paper, we use DARPA training dataset with our training module and DARPA evaluation dataset as our test audit data to measure the effectiveness of the resulting NIDS rules.

To evaluate AceGA and its enhancement features across multiple perspectives, four experiments were performed. We first test the fitness score of the wildcards-filled individual to show how this form of individual can be mistakenly regarded as a top-ranked rule. Consequently, it demonstrates how our Wildcard Weight Penalty elegantly eliminates this drawback. Our second experiment evaluates the time performance of AceGA when compared against NetGA [1]. The experiment also demonstrates how our Elite's Traits Induction enhancement can accelerate the training process. Next, Ace Comparison Elimination experiment investigates how this enhancement of AceGA increases the variety of the generated rules and solves the over-fitting problem. Our last simulation experiment reveals the quality of NIDS rules produced by AceGA when used to detect our four selected attacks. The numbers of true positive, true negative, false positive, and false negative are shown along with the resulting accuracy, precision, sensitivity, and specificity rates.

This rest of this paper is organized as follows. Section 2 provides some essential background and related research contributions. Our proposed AceGA scheme is detailed in Section 3. Section 4 outlines the experimental evaluation of AceGA. Some further discussion is presented in Section 5. Finally, Section 6 draws the paper to conclusion and projects some promising future work.

2. Background

Genetic algorithms (GAs) are heuristic search algorithms based on the genetic processes, which natural populations evolve according to the principle of natural selection. GA mimics this process to evolve solutions, mostly to search and optimization problems. The group of individuals (i.e. solutions to the problem) that evolves throughout the GA evolutionary process is called *population*. An individual must be represented as a *chromosome* structure (or genome) [9, 10], which provides a blueprint for its creation. A chromosome consists of a number of genes, each of which has its fixed position in the chromosome called a *locus*. Each genetic locus stores a *trait* that represents specific meaningful information about the individual. During the creation of an individual, the trait at each locus would be randomly chosen from the pool of possible values for that specific locus, called *alleles*. After the population of solutions is created, genetic operators such as selection, crossover, and mutation are applied to evolve and generate the solutions. In each evolution, fitness function evaluates every genetic individual to determine the likeliness that it can survive the evolution.

A number of studies have been conducted over the past 15 years in an attempt to integrate Genetic Algorithms into NIDS to overcome the limitations of anomaly-based and signature-based IDS. An early research on GAs for IDS [18] involves the application of multiple agent technology and Genetic Programming to detect network anomalies. Each agent monitors one network parameter of the audit

data, while Genetic Programming is used to find a set of agents that collectively determine network behavior anomalies. However, this approach is proved to be time consuming. To study the feasible use of GA in model generation for rule-based IDS, a representation of network connection and its related behavior as IDS rules was proposed [19]. The resulting set of rules can be used to determine whether or not a real-time connection is intrusive. The proposed scheme calculates the fitness value based on the number of connections that are grouped as attacks and the number of attacks correctly detected. Li [13] offered an approach to use GA in anomaly-based IDS and provided a blueprint chromosome structure for some later research attempts [1, 4]. The work also suggested the use of different weights for different network features in fitness evaluation. Gong et al. [4] later partially demonstrated how Li's approach could be realized. The proposed scheme represents an individual rule by 'if A then B' and adapts a support-confidence framework fitness function [4, 20, 21] to calculate the fitness values. Based on the works of Li [13] and Gong et al. [4], NetGA [1] was developed, providing a detailed guideline on the selection, crossover, and mutation operators for the GA process. The proposed scheme also successfully integrated NetGA into nProbe network monitoring software [34]. An IDS environment was created to test the generated rules with a DARPA test audit data file. While NetGA meets its functional requirement and can generate local maximal rules, it still lacks in extensibility. It was modeled to run only on one sample of DARPA audit training and testing dataset [11]. Moreover, its data training process is also relatively time-consuming. Instead of using GA to directly generate NIDS rules, some other research [22, 23] utilizes GA for features selection and integrates it with other AI techniques, including fuzzy techniques [6, 24-26], artificial neural network [27, 28], support vector machine [29, 30], and decision tree classifier [31], to derive NIDS rules.

Our AceGA applies GA in NIDS rules generation while addressing some limitations of the existing research efforts [1, 4, 13]. It involves multiple enhancements both in the GA process and the fitness function, as presented in Section 3-5.

3. Design and Implementation of AceGA

The use of genetic algorithms usually concerns three main aspects, namely the genetic representation, the definition of fitness function, and the definition and implementation of genetic algorithm process. This section describes the design AceGA in these three facets.

3.1 Genetic Representation and AceGA's Population

Based on the preliminaries in Section 2, each individual must appear as a chromosome. AceGA adopts a previously proposed chromosome structure [1, 4], which is in the form of one-dimensional list of fifteen elements. They represent fifteen genetic loci, which store six network features considered by our NIDS and the corresponding attack type. Table 1 outlines the features used to construct our chromosome.

Feature	Format	Number of Genes
Duration	h:m:s	3
Protocol	Int	1
Source_port	Int	1
Destination_port	Int	1
Source_IP	a.b.c.d	4
Destination_IP	a.b.c.d	4
Attack_name	Int	1

Table 1: Selected Network Features

AceGA's population are produced from DARPA network session training audit dataset, which contain seven network features and some auxiliary data. The data are extracted and arranged into fifteen elements. They signify the genes at the fifteen genetic loci of the chromosome as follows.

[h, m, s, protocol, src_port, dest_port, 1st_Octet_srcIP, 2nd_Octet_srcIP, 3rd_Octet_srcIP, 4th_Octet_srcIP, 1st_Octet_destIP, 2nd_Octet_ destIP, 3rd_Octet_destIP, 4th_Octet_ destIP, attack_name]#

Table 2 provides three sample chromosomes produced from three DARPA training data records. The first one shows that if a network packet originates from IP address 192.168.1.30 and port 1922, and is sent to IP address 192.168.0.20 and with port 1021 using the rsh protocol, and the connection duration is 2 seconds, then it is highly possible that the network connection is an attack of type 'rsh'. Likewise, the third one denotes a connection if a network packet comes from IP address 192.168.0.40 and port 4356, and is delivered to IP address 192.168.0.20 and with port 23 using the telnet protocol, and the connection duration is 1 minute and 24 seconds, then it is highly likely that the network connection is a normal connection.

```
[0, 0, 2, 'rsh', 1022, 1021, 192, 168, 1, 30, 192, 168, 0, 20, 'rsh']
[0, 0, 40, 'telnet', 1914, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
[0, 1, 24, 'telnet', 4356, 23, 192, 168, 0, 40, 192, 168, 0, 20, '-']#
```

Table 2: Samples of training data arranged in chromosome structure

Our population are derived from these training data's chromosomes. All of the available values at each genetic locus of all chromosomes are extracted, with duplications removed, to produce the allele containing unique possible traits at that particular locus. Additionally, a wildcard symbolized by -1 is added to every allele, except for the allele of the last locus that identifies the attack name. As an example, based on the training data in Table 2, the allele of the first locus becomes {0, -1}, that of the second locus is {0, 1, -1}, the one for the fourth locus gets to be {'rsh', 'telnet', -1}, whereas the allele of the last locus is {'rsh', 'guess'}.

To generate an individual of the population, the trait at each locus is randomly chosen from the values in its corresponding allele. These individuals actually are NIDS rules in development. For each locus, the probability for the wildcard to be selected, i.e. Wildcard Weight (WCW), is adjustable from 0.0 to 1.0. The probability for other traits to be selected (PB_{trait}) is uniformly distributed, namely:

$$PB_{trait} = (1 - WCW) / (Number\ of\ all\ traits - 1)$$

From the samples in Table 2 and the resulting alleles, a possible genetic individual is [0, 0, 40, 'rsh', 4356, 1021, 192, 168, 1, -1, 192, 168, 0, 20, 'rsh']. Each individual depicts an if-then clause, where the first 14 genes portray the network connection rule's conditions connected by logical AND operations, while the last gene is its outcome. Hence, the individual example above depicts the rule *"if (duration='0:1:14' and protocol='rsh' and source_port=4356 and destination_port=1021 and source_ip='192.168.1.*' and destination_ip='192.168.0.20') then (attack_name='rsh')"*.

3.2 Fitness Function and Wildcard Penalty Enhancement

The more wildcards in NIDS rules, the more connections would be denied, increasing the chance of false positive detection. Manifestly, the rule fully filled with only wildcards must be avoided since it blocks all the connections, rendering the network inaccessible. Unfortunately, traditional support-confidence framework fitness function [1, 4, 20, 21] disregards this occurrence and gives individuals with multiple wildcards comparatively high fitness values. To remedy this particular issue, AceGA introduces Wildcard Penalty Enhancement, which includes some penalty in the function as depicted in Figure 1. It deducts from individuals containing wildcards a small portion of fitness value equivalent to the number of wildcards multiplied by a configurable penalty weight, $W_{wc\_penalty}$. This enhancement corrects the misconception that an individual filled with wildcards should have high fitness value, making the function more applicable to NIDS arena. In other words, the individuals with less precisely matched genetic composition would have lesser fitness. Analogically, our scheme creates a competition between individuals to change their genetic compositions to achieve higher fitness values and survive the evolution. Thus, it ensures that the individuals are evolved to become better with every generation. An experiment evaluating this enhancement is presented in Section 4.1.

$$penalty = Wildcard\ Penalty\ Weight \times Number\ of\ wildcards$$
$$support = |A\ and\ B| / N$$
$$confidence = |A\ and\ B| / |A|$$
$$fitness = (w1 \times support) + (w2 \times confidence) - penalty$$

Figure 1: Fitness function with Wildcard Penalty Enhancement

Consider our support-confidence fitness function with wildcard penalty enhancement. *N* is the total of number of network connection rules in the training dataset. |*A*| indicates the number of rules whose condition parts (i.e. the first 14 elements) match the condition A. |*A and B*| is the number of network connection rules whose condition parts match the condition A and outcome parts (i.e. the last element) match the outcome B. The weights *w1* and *w2*, whose values sum to one, control the balance between the support and the confidence, without crucial effect on the performance [4]. The greater the value of *w1*, the simpler it is to identify network intrusions. On the contrary, higher *w2* leads to more precise classification of the intrusions. The default values of *w1* and *w2* are 0.2 and 0.8 respectively [4].

idx	Duration			Protocol	SRC Port	DST Port	SRC IP				DST IP				Attk Type
1	0	0	0	smtp	1900	25	192	168	1	30	192	168	0	20	-
2	0	0	2	rsh	1023	1021	192	168	1	30	192	168	0	20	rcp
3	0	0	23	telnet	1906	23	192	168	1	30	192	168	0	20	guess
4	0	0	14	rlogin	1022	513	192	168	1	30	192	168	0	20	rlogin
5	0	0	2	rsh	1022	1021	192	168	1	30	192	168	0	20	rsh
6	0	0	15	ftp	4354	21	192	168	0	40	192	168	0	20	-
7	0	0	40	telnet	1914	23	192	168	1	30	192	168	0	20	guess
8	0	1	24	telnet	4356	23	192	168	0	40	192	168	0	20	-
Individual															
x	0	0	2	rsh	-1	-1	192	168	1	30	192	168	-1	20	rsh

Table 3: Example of an individual rule x matching against a training dataset

To exemplify the use of our fitness function, consider a network training dataset and an individual rule x in Table 3. The condition parts of the rule 2 and 5 in the training dataset (i.e. their first 14 attributes) match that of our individual. However, only the rule 5 has the outcome identical to the individual's. As a result, using default *w1* and *w2*, and $W_{wc\_penalty}$ of 10^{-12}, the fitness value for the individual x can be calculated in the following way:

N = 8 connections, |A| = 2, |A and B| = 1, Number of Wildcards = 3
penalty = $W_{wc\_penalty}$ x 3 = 3 x 10^{-12}
support = |A and B| / N = 1/8 = 0.125
confidence = |A and B| / |A| = 1/2 = 0.5
fitness value = (0.2 x 0.125) + (0.8 x 0.5) - (3 x 10^{-12}) = 0.424999999997

3.3 Evolutionary Process

As illustrated in Figure 2, the genetic algorithm evolutionary process of AceGA is relatively straightforward. Once a population of individuals are initialized as described in Section 3.1, they enter the evolutionary loop for a certain amount of generations and evolve to ultimate individuals.

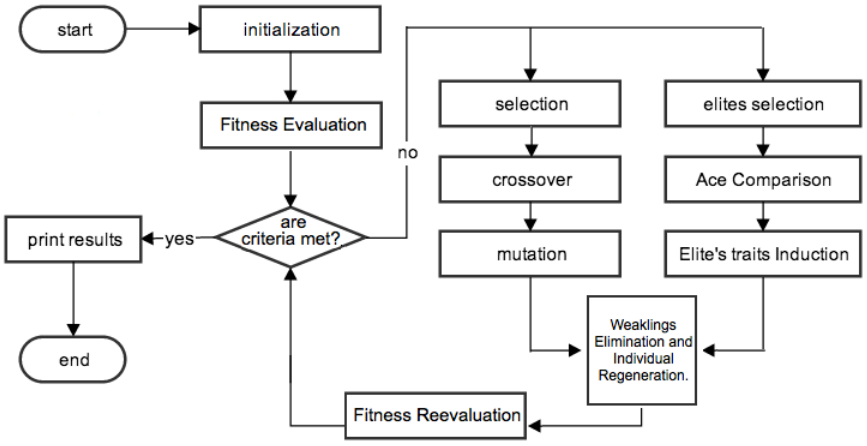

Figure 2: AceGA Training Module diagram

For each evolutionary loop, individuals in the current population progress through two courses of process. The first one follows the classical GA process. On the second route, AceGA engages GA's Elitism process, which is a hill climbing method that reserves space for a certain number of the enhanced individuals of each generation in the offspring. Through Elitism, the enhanced fitness is guaranteed since it ensures that the individuals with highest fitness values remain unmodified and survive the evolutions. As these elites are not removed from the population, they continue to produce offspring with existing high quality features. Our proposed Ace Comparison and Elite's traits Induction enhancements are applied to the Elitism process as will be elaborated in Section 3.3.2. Towards the end of the evolutionary loop, AceGA incorporates an additional step called Weaklings Elimination to accommodate the effect of our enhanced Elitism process and regenerate the new individuals, as detailed in Section 3.3.3. Finally, the fitness values of all the prevailing individuals are reevaluated before they assigned as the population for the next evolutionary loop. The individuals continue to evolve until the number of evolutionary loops exceeds the predefined value. At the completion, the survived population become the NIDS rules.

3.3.1 Classical GA Operations

During the selection operation, a predefined number of individuals, normally 100% or 75% of the population, are randomly selected. These individuals are then passed on to the crossover operation, where individuals are randomly chosen, one pair at a time, as parents to mate. The probability for a pair to crossover is definable and is set to 0.8 by default. AceGA utilizes two-points crossover operation. Here, two points of reference among the 15 network features are randomly selected. Next, all genetic traits within that particular range are exchanged between the parents, returning two children from each pair of parents. An example of crossover operation is displayed in Table 4.

Mutation operation involves altering one or more gene values of the individuals. The likelihood for an individual to undergo mutation is determined by a definable mutation probability. For AceGA, the gene at every genetic locus is given a probability of 0.25 to be replaced by another value taken from the allele corresponding to that locus. Table 5 demonstrates an instance of our mutation operation.

idx	Duration			Protocol	SRC Port	DST Port	SRC IP				DST IP				Attk Type
Parents															
1	0	1	2	ssh	7665	22	114	55	8	70	192	168	0	20	pod
2	0	0	23	telnet	1906	23	10	0	1	5	172	168	1	1	Guess
Children															
1	0	1	2	ssh	7665	22	114	0	1	5	172	168	0	20	pod
2	0	0	23	telnet	1906	23	10	55	8	70	192	168	1	1	guess

Table 4: Crossover operation being performed on two parents resulting in two children

idx	Duration			Protocol	SRC Port	DST Port	SRC IP				DST IP				Attk Type
1	0	0	0	smtp	1900	25	192	168	1	30	192	168	0	20	guess
			↓							↓					↓
1	0	0	2	smtp	1900	25	192	168	1	40	192	168	0	20	rcp

Table 5: Mutation operation performed on an individual

3.3.2 Elitism Process

Elites Selection Operation

Elites selection categorizes every individual in the current population by their attack types. Then a definable number of non-duplicate top individuals (elites) with highest fitness values are selected; each of which is given a reserved spot in the offspring population.

Ace Comparison Elimination Enhancement

Existing GANIDS approaches [1, 4] suffer the local maxima over-fitting problem, leading to the lack in the variety of NIDS rules being generated. This problem exists because the individuals always evolve themselves following the pattern of already existing local maxima during each generation.

Accordingly, only one type of local maximum starts to overrun the population of the elites, impeding the development of other candidates with relatively lesser fitness values but are potentially useful.

Ace Comparison Elimination enhancement resolves this issue by saving only one spot per attack type, called Ace, for the local maximal elites. Other elites with similar fitness values are eliminated from the elite caste, allowing more space for other potential candidates to evolve as NIDS rules.

The enhancement is implemented as a function that takes a list of all elites as its inputs. Then it classifies each elite into groups according to their attack types. From every group, the elite with the highest fitness value is selected and stored as an Ace, which is later used in a comparison process called fitness comparison. The fitness comparison compares the fitness value of every other elite against that of the Ace. If the fitness difference is lower than the predefined threshold value, the elite is eliminated from the elite space. In other words, the threshold, which ranges from 0.0 to 1.0, defines line at which fitness difference between Ace and the elite being considered becomes insignificant. After the comparison process, the function returns a list of the remaining elites. Our experiment shows that AceGA has put forward a genetic algorithm that is able to generate diverse IDS rules containing the local maxima as well as other potentially useful candidates.

Elite's Traits Induction Enhancement

Foregoing efforts on GANIDS [1, 4] generate random individuals with low wildcard weight or probability of containing wildcards in their genes (e.g. 0.1). However, we discovered that it is possible to accelerate the evolutionary process using a reverse concept, which applies high wildcard weight (e.g. 0.9). When most of the traits being generated in the individuals become wildcards, such as *[0, -1, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 20, 'port-scan',* such individuals can converge toward high fitness value more rapidly. Inevitably, this technique creates individuals with excessive wildcards, which are unpleasant as discussed in Section 3.2. Nonetheless, it is designed to be used in conjunction with Wildcard Penalty enhancement. When working hand in hand, they can neatly increase the performance of the GANIDS while achieving desirable individuals.

Starting from elite individuals containing a multitude of wildcards, Elite's Traits Induction enhancement function changes the existing wildcards to more specific values as offered in the alleles. It encompasses several steps. To begin with, taking all of the elite individuals as its input, it processes each one at a time. For every elite, the function searches through its loci for those storing a wildcard value. Among the identified loci, exactly one locus is randomly selected. Next, with mutation probability, the trait at the selected locus mutates to one of the traits in the allele belonging to that locus. If the resulting elite varies in term of genetic composition from the input elite, it is then copied over to the offspring population.

Elite's Traits Induction enhancement enables AceGA to increase the fitness of the elites while decreasing their convergence times by up to the factor of 8 when compared to the existing approach that waits until the traits are properly allocated together before the individuals can start to converge toward the desired fitness. The comparison between different wildcard weight settings and the effects of *Elite's Traits Induction* enhancement are shown in Section 4.2.

3.3.3 Weaklings Elimination

Since an uncertain number of elites and mutated elites have been added to the offspring population, Weakling Elimination removes some individuals to keep the population size balanced to that at the initialization stage. As a side benefit, this process also provides an opportunity to remove individuals with low fitness in order to offer spaces for new individuals to be recreated. The regeneration of such individuals brings more diverse genetic traits into the population, which in turn expedites the discovery of global maxima.

AceGA eliminates a definable number of weaklings, $N_{base\_weaklings}$ before generating new individuals. The overall steps are outlined below, where N_{elite} is the number of elites, $N_{mutated\_elite}$ denotes the amount of mutated elites produced during Elite's Traits Induction, N_{pop} is the current population size, and $N_{offspring}$ indicates the size of the offspring.

1. Determine the number of individuals to be removed, $N_{remove} = N_{base\_weaklings} + N_{elite} + N_{mutated\_elite}$
2. Remove N_{remove} weakest individuals from the offspring population
3. Calculate the number of spaces available for new individuals, $N_{avail} = N_{pop} + N_{offspring}$
4. Regenerate N_{avail} new individuals and add them to the offspring

4. Evaluation and Results

AceGA* algorithm is implemented in Python 2.7 together with DEAP [16] library, interpreted and run on PyPy [17] for better speed. In this section, a series of simulation experiments is conducted to rigorously evaluate different enhancements included in AceGA. All of them are performed on a system with 2.5 GHz CPU and 4 GB of memory running Mac OSX Lion 10.7.

4.1 Wildcard Penalty Experiment

As discussed in length in Section 3.2, rules with excessive number of wildcards are highly likely to block legitimate connections and should be avoided. This section shows that the application of wildcard penalty enhancement since its initialization enables AceGA to assign higher fitness to individuals with less wildcard traits which in turn allows the individual to converge correctly.

Our first experiment compares the solutions generated by AceGA with NetGA [1] which uses traditional fitness function [4, 20, 21]. Both of the schemes are trained with DARPA training sample data bsm.list [11] using the same genetic algorithm parameters as shown in Table 6.

Simulation Parameters for Elite's Traits Elimination Experiment and Performance Evaluation	
Size of Population	400
Number of individuals selected in selection operation	100% of population
Maximum number of evolutionary loops	400
Number of elites per attack type	2
Wildcard weight for NetGA and AceGA	0.1
Wildcard weight for AceGA with Elite's Traits Induction enhancement	0.9
Wildcard penalty weight	0.9

Table 6: Simulation Parameters for Performance Evaluation

Outcomes from NetGA	
Run 1:	*fitness: 0.8063 [0, 0, 5, -1, -1, -1, -1, -1, 1, 30, 192, 168, 0, 20, 'port-scan']* *fitness: 0.8063 [0, 0, 23, 'telnet', -1, -1, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']*
Run 2:	*fitness: 0.8063 [0, 0, 5, -1, -1, -1, 192, 168, 1, 30, 192, 168, 0, 20, 'port-scan']* *fitness: 0.8063 [0, -1, 23, -1, -1, -1, 192, 168, -1, 30, 192, 168, 0, 20, 'guess']*
Run 3:	*fitness: 0.8063 [0, 0, 5, -1, -1, -1, 192, 168, 1, -1, 192, 168, 0, 20, 'port-scan']* *fitness: 0.8063 [0, 0, 23, 'telnet', -1, -1, -1, 168, 1, 30, 192, 168, 0, 20, 'guess']*
Outcomes from AceGA	
All Runs:	*fitness: 0.80625 [0, 0, 5, -1, -1, -1, 192, 168, 1, 30, 192, 168, 0, 20, 'port-scan']* *fitness: 0.80625 [0, 0, 23, 'telnet', -1, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']*

Table 7: Solutions for bsm.list DARPA training dataset discovered by NetGA and AceGA

Table 7 outlines the NIDS rules produced for 'guess' and 'port-scan' attack types by NetGA and AceGA. Noticeably, the solutions produced by NetGA vary in term of genetic composition across different runs regardless of the number of evolutionary loops. Although they show some resemblance, it is still difficult to verify which ones are correct. Certainly, they cannot be used in conjunction, as that would block too many legitimate network connections. On the contrary, those generated by AceGA are completely stable across all runs and always returns the less number of wildcards.

4.2 Elite's Traits Induction Experimental Study and Performance Evaluation

This experiment not only evaluates the time performance of AceGA when compared against NetGA, but also demonstrates how our Elite's Traits Induction enhancement can accelerate the training process. We use the training dataset and simulation parameters identical to the experiment in Section 4.1. Thirty rounds of simulation are conducted to acquire the average result.

The results from UNIX time command in Table 8 reveal that AceGA even without Elite's Traits Induction enabled consumes approximately 46% less time than NetGA. Moreover, our Elite's Traits Induction enhancement can further decrease AceGA's execution time by almost 75%. Note that with our enhancement, only 60 evolutionary loops are required to discover the global maxima. In other words, it can reach global maxima faster than NetGA and AceGA without Elite's Traits Induction by the factor of 8 and 4 respectively.

* The source code of AceGA is available at https://github.com/nixor/GANIDS

Time	NetGA	AceGA	AceGA with Elite's traits Induction enhancement
Execution Time (real)	25.866 s	11.992 s	3.113 s
CPU time in user-mode code (user)	25.729 s	11.825 s	3.052 s
CPU time in kernel (sys)	0.106 s	0.074 s	0.052 s

Table 8: Training time comparison

4.3 Ace Comparison Elimination Experiment

The advantage of Ace Comparison Elimination can be demonstrated by observing the difference in the diversity of rules generated by the AceGA in two scenarios; one with Ace Comparison Elimination disabled, and the other with it enabled. The parameters used in this simulation are presented in Table 9.

Simulation Parameters for Elite's Traits Elimination Experiment and Performance Evaluation	
Size of Population	800
Number of evolutionary loops	200
Wildcard weight	0.9
Mutation probability	0.9
Ace Comparison Elimination fitness difference threshold	10^{-3}

Table 9: Simulation Parameters for Ace Comparison Elimination Experiment

Trained with DARPA training dataset, the resulting NIDS rules from AceGA with Ace Comparison Elimination disabled and enabled are shown in Figure 3 and Figure 4 respectively. Evidently, Ace Comparison Elimination enhancement offers are richer NIDS rules in term of the variety of genetic compositions. The global maxima are conserved, while the other maxima are allowed to develop as well. Therefore, this enhancement can solve the over-fitting problem [4].

fitness: 0.80624999993600 [0, 0, 23, 'telnet', -1, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
fitness: 0.80624999987200 [0, 0, 23, 'telnet', -1, 23, 192, 168, 1, -1, 192, 168, 0, 20, 'guess']
fitness: 0.80624999987200 [0, -1, 23, 'telnet', -1, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
fitness: 0.80624999987200 [-1, 0, 23, 'telnet', -1, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
fitness: 0.80624999987200 [0, 0, 23, 'telnet', -1, 23, 192, 168, 1, 30, -1, 168, 0, 20, 'guess']
fitness: 0.80624999980800 [0, 0, 5, -1, -1, -1, 192, 168, 1, 30, 192, 168, 0, 20, 'port-scan']
fitness: 0.80624999974400 [0, 0, 5, -1, -1, -1, 192, 168, 1, 30, 192, 168, -1, 20, 'port-scan']
fitness: 0.80624999974400 [0, 0, 5, -1, -1, -1, 192, 168, -1, 30, 192, 168, 0, 20, 'port-scan']
fitness: 0.80624999974400 [0, 0, 5, -1, -1, -1, -1, 168, 1, 30, 192, 168, 0, 20, 'port-scan']
fitness: 0.80624999974400 [0, 0, 5, -1, -1, -1, 192, 168, 1, 30, 192, 168, 0, -1, 'port-scan']

Figure 3: Rules generated with Ace Comparison Elimination disabled

fitness: 0.80624999993600 [0, 0, 23, 'telnet', -1, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
fitness: 0.80312500000000 [0, 0, 23, 'telnet', 1906, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
fitness: 0.80312500000000 [0, 0, 22, 'telnet', 1867, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
fitness: 0.80312500000000 [0, 0, 23, 'telnet', 1884, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
fitness: 0.80312499993600 [0, 0, -1, 'telnet', 1867, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'guess']
fitness: 0.80624999980800 [0, 0, 5, -1, -1, -1, 192, 168, 1, 30, 192, 168, 0, 20, 'port-scan']
fitness: 0.80312499993600 [0, 0, 3, 'finger', 2023, 79, 192, 168, 1, -1, 192, 168, 0, 20, 'port-scan']
fitness: 0.80312499993600 [0, 0, 5, 'exec', -1, 512, 192, 168, 1, 30, 192, 168, 0, 20, 'port-scan']
fitness: 0.80312499993600 [0, 0, 3, -1, 2023, 79, 192, 168, 1, 30, 192, 168, 0, 20, 'port-scan']
fitness: 0.80312499993600 [0, 0, 5, 'telnet', -1, 23, 192, 168, 1, 30, 192, 168, 0, 20, 'port-scan']

Figure 4: Rules generated with Ace Comparison Elimination enabled

4.4 NIDS Rules Quality Evaluation

Our last experiment simulates the test module to apply the generated NIDS rules against the DARPA test audit data to verify the quality of the rules produced from the AceGA's training module, as listed in Table 10 below. Four out of twenty seven available types of attacks have been chosen as sample inputs for this study. After AceGA has been trained with the training audit data obtained across the seven weeks of the DARPA network-monitoring period [15], the produced rules are then passed to the test module. Each attack type simulation is run against 1,542,781 network sessions in the DARPA test audit data to classify which of them are likely to be attack connections. Finally, our outcomes are compared with the real attack schedules provided on the DARPA website [11], which details the actual types and times of the attacks in the test audit data.

pod	Denial of service ping of death
teardrop	Denial of service where mis-fragmented UDP packets cause some systems to reboot.
neptune	Syn flood denial of service on one or more ports.
ipsweep	Surveillance sweep performing either a port sweep or ping on multiple host addresses.

Table 10: Attacks chosen for test module simulation and their descriptions

The overall results and their related significant statistics are listed in Table 11 and Table 12. It can be observed that the NIDS rules trained by AceGA can competently detect all of the chosen attack types, with high accuracy, precision, sensitivity, and specificity measurement rates. Similar to other signature-based NIDS, the capability of AceGA depends on multiple factors as discussed in Section 5.

Attack Type	True Positive	False Positive	True Negative	False Negative
pod	88.3166%	11.6834%	100%	0%
Teardrop	99.8752%	0.1248%	100%	0%
Neptune	94.7335%	5.2665%	100%	0%
ipsweep	72.6708%	27.3292%	100%	0%

Table 11: Test module simulation classification results

Attack	True Positive Rate	False Positive Rate	Accuracy	Precision	Sensitivity	Specificity
pod	1.0	0.1046	99.9841%	88.3166%	100%	99.9841%
Teardrop	1.0	0.0012	100%	99.8860%	100%	100%
neptune	1.0	0.2146	99.2526%	94.7335%	100%	99.1366%
ipsweep	1.0	0.0500	99.9258%	94.7335%	100%	99.1366%

Table 12: Test module simulation results' statistics

5. Discussion

5.1 NIDS Rules Quality

Table 11 and Table 12 in Section 4.4 show that the rules generated by AceGA when trained with DARPA training audit dataset [15] can proficiently detect all of the chosen attack types in the test audit dataset, namely pod, teardrop, nepture and ipsweep. Nevertheless, we still cannot guarantee of its ability to detect all of the real-world attacks and require further investigation. Both the GA algorithm used in the training and the training dataset contribute to the detection capability of GANIDS. In addition, while it cannot be ensured that merely seven network features are sufficient for training AceGA to effectively detect a broader array of network attacks, the design of AceGA is flexible enough to accommodate more network features. Lastly, whereas AceGA is experimented with DARPA training and test dataset, there are other datasets available such as KDDCUP [32] and ISCX datasets [33]. Thus, there is still room for future experiment on AceGA.

5.2 Time Performance

The training speed is multiple influenced by multiple factors including the implementation of the genetic algorithm, the hardware platform, and the training and test dataset. In our study, we developed and evaluated AceGA on a personal computer. Undoubtedly, AceGA shall achieve markedly better training speed on a high-end platform. Moreover, our implementation of AceGA algorithm along with its elitism enhancements is a prototype at the current stage. The proposed algorithm can be programmed in other languages with different program structures to improve its execution time.

5.3 The limitations of AceGA

Even though AceGA solves the over-fitting problem [4] by introducing Ace Comparison Elimination enhancement to conserve both the best local maxima and the second best local maxima, it does not guarantee the conservation of lower fitness but potential individuals such as the third or fourth best global maxima. This is moderately difficult to accomplish because the fitness differences of the global maxima are different for each set of the training data. In some scenarios, the fitness of the enhanced individuals could be as high as 0.99, while it could be as low as 0.60 in some others. A pertinent prediction of these patterns is required to address this limitation.

The process of selecting top individuals trained by the training module is still manually handled. As mentioned earlier, while AceGA can greatly facilitate in NIDS rule generation process, it is not yet designed to fully replace security experts. Lastly, any GANIDS software creates new IDS rules from existing patterns of attacks, and so does the AceGA. New attacks that do not derive from the existing patterns cannot be prevented using this GANIDS approach.

6. Conclusion and Future Work

In response to an explosive number of network attacks, we observe numerous ongoing research efforts aiming to develop efficient and effective NIDS, including the integration of Artificial Intelligence such as Genetic Algorithms, known as GANIDS. In this paper, we provide insights into this multi-disciplinary research area along with existing limitations, and propose a promising GANIDS approach called AceGA. Through an extensive series of experiments, AceGA together with its three novel enhancements has been proved to produce high-quality NIDS rules in a more efficient manner. The Wildcard Weight Penalty enhancement has addressed the issue concerning wildcards filled individuals to ensure that the actual global maxima are found in the evolution. The training time of AceGA is about 50% of that of the existing scheme. Besides, the Elite's Traits Induction enhancement can further improve the rules generation time by the factor of 4 on average. Lastly, the Ace Comparison Elimination enhancement remedies the existing maxima over-fitting problem and increases the diversity of the resulting NIDS rules.

There are several avenues for future research. First, we will investigate the potential of integrating semantics of the network features into the GA process to develop a more intelligent GANIDS. Second, we plan to incorporate other Artificial Intelligence techniques such as Neural Network into AceGA to improve the generated rules quality. Finally, we will pioneer a technique to exploit our Elite's Traits Induction scheme in other genetic algorithm-based systems to speed up the evolutionary process.

7. References

[1] B. E. Lavender and S. California State University, Implementation of Genetic Algorithms Into a Network Intrusion Detection System (netGA), and Integration Into NProbe: California State University, Sacramento, 2010.

[2] M. S. Hoque, M. Mukit, M. Bikas, and A. Naser, "An implementation of intrusion detection system using Genetic Algorithm," arXiv preprint arXiv:1204.1336, 2012.

[3] B. Shah and B. H. Trivedi, "Artificial Neural Network based Intrusion Detection System: A Survey," International Journal of Computer Applications, vol. 39, 2012.

[4] R. H. Gong, M. Zulkernine, and P. Abolmaesumi, "A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection," presented at the Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks, 2005.

[5] SANS Security. "SANS: Intrusion Detection FAQ: Statistical based approach to Intrusion Detection". [Online]. Available: http://www.sans.org/security-resources/idfaq/statistic_ids.php [Accessed: Mar. 6, 2013].

[6] S. M. Bridges and R. B. Vaughn, "Fuzzy data mining and genetic algorithms applied to intrusion detection," in Proceedings 23rd National Information Security Conference, 2000.

[7] B. Abdullah, I. Abd-alghafar, G. I. Salama, and A. Abd-alhafez, "Performance Evaluation of a Genetic Algorithm Based Approach to Network Intrusion Detection System," in 13th International Conference on Aerospace Sciences & Aviation Technology, 2009, pp. 1-17.

[8] F. A. Anifowose and S. I. Eludiora, "Application of Artificial Intelligence in Network Intrusion Detection," 2011.

[9] M. Mitchell, "An introduction to genetic algorithms, 1996," PHI Pvt. Ltd., New Delhi, 1996.

[10] S. Sivanandam and S. Deepa, Introduction to genetic algorithms: Springer Publishing Company, Incorporated, 2007.

[11] MIT Lincoln Laboratory. "Communication Systems and Cyber Security: Cyber Systems and Technology: DARPA Intrusion Detection Evaluation". [Online]. Available: http://www.ll.mit.edu/mission/communications/cyber/ CSTcorpora/ideval/data/1998data.html [Accessed: Apr. 20, 2013].

[12] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, et al., "Evaluating intrusion detection systems without attacking your friends: The 1998 DARPA intrusion detection evaluation," DTIC Document1999.
[13] W. Li, "Using genetic algorithm for network intrusion detection," in Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference, Kansas City, Kansas, 2004, pp. 24-27.
[14] A. Ojugo, A. Eboka, O. Okonta, and F. Aghware, "Genetic Algorithm Rule-Based Intrusion Detection System (GAIDS)," Journal of Emerging Trends in Computing and Information Sciences, vol. 3, 2012.
[15] C. Thomas, V. Sharma, and N. Balakrishnan, "Usefulness of DARPA dataset for intrusion detection system evaluation," in SPIE Defense and Security Symposium, 2008.
[16] deap. "Distributed Evolutionary Algorithms in Python - Google Project Hosting". [Online] Available: https://code.google.com/p/deap/ [Access: Mar. 26, 2013].
[17] PyPy. "Download and install". [Online]. Available: http://pypy.org/ [Accessed: Mar. 26, 13].
[18] M. Crosbie and G. Spafford, "Applying genetic programming to intrusion detection," in Working Notes for the AAAI Symposium on Genetic Programming, 1995, pp. 1-8.
[19] A. Chittur, "Model generation for an intrusion detection system using genetic algorithms," Honors Thesis, Ossining High School. In cooperation with Columbia Univ., 2001.
[20] W. Lu and I. Traore, "Detecting new forms of network intrusion using genetic programming," Computational Intelligence, vol. 20, pp. 475-494, 2004.
[21] M. L. Wong and K. S. Leung, Data mining using grammar based genetic programming and applications: Kluwer Academic, 2000.
[22] S. Zaman, M. El-Abed, and F. Karray, "Features selection approaches for intrusion detection systems based on evolution algorithms," in Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, 2013, p. 10.
[23] M. Middlemiss and G. Dick, "Feature selection of intrusion detection data using a hybrid genetic algorithm/KNN approach," in Design and application of hybrid intelligent systems, A. Ajith, K. Mario, ppen, and F. Katrin, Eds., ed: IOS Press, 2003, pp. 519-527.
[24] C.-H. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection," Pattern Recognition, vol. 40, pp. 2373-2391, 2007.
[25] M. S. Abadeh, J. Habibi, and C. Lucas, "Intrusion detection using a fuzzy genetics-based learning algorithm," Journal of Network and Computer Applications, vol. 30, 2007.
[26] T. P. Fries, "A fuzzy-genetic approach to network intrusion detection," in Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation, 2008.
[27] A. Gomathy and B. Lakshmipathi, "Network Intrusion Detection Using Genetic Algorithm and Neural Network," in Advances in Computing and Information Technology, ed: Springer, 2011, pp. 399-408.
[28] G. Kumar and K. Kumar, "The Use of Multi-Objective Genetic Algorithm Based Approach to Create Ensemble of ANN for Intrusion Detection," International Journal of Intelligence Science, vol. 2, pp. 115-127, 2012.
[29] S. Saha, A. S. Sairam, A. Yadav, and A. Ekbal, "Genetic algorithm combined with support vector machine for building an intrusion detection system," in Proceedings of the International Conference on Advances in Computing, Communications and Informatics, 2012.
[30] P. Sujatha, C. S. Priya, and A. Kannan, "Network intrusion detection system using genetic network programming with support vector machine," in Proceedings of the International Conference on Advances in Computing, Communications and Informatics, 2012, pp. 645-649.
[31] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, "Decision tree classifier for network intrusion detection with GA-based feature selection," in Proceedings of the 43rd annual Southeast regional conference-Volume 2, 2005, pp. 136-141.
[32] The UCI KDD Archive. "KDD Cup 1999 Data". [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Accessed: Mar. 27, 2013].
[33] Information Security Center of eXcellence. "UNB ISCX Intrusion Detection Evaluation DataSet". [Online]. Available: http://www.iscx.ca/datasets [Accessed: May. 17, 2013].
[34] nprobe. "An Extensible NetFlow v5/v9/IPFIX GPL Probe for IPv4/v6 ". [Online] Available: http://www.ntop.org/products/nprobe/ [Access: Mar. 28, 2013].